Docket No. 42P17508
Express Mail No. EV339916799US

UNITED STATES PATENT APPLICATION

FOR

ACCESSING CONFIGURATION REGISTERS BY AUTOMATICALLY CHANGING AN INDEX

Inventors:

Tuan M. Quach
Reza E. Daftari

Prepared by:
BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
12400 Wilshire Boulevard, Seventh Floor
Los Angeles, California 90025
(310) 207-3800

ACCESSING CONFIGURATION REGISTERS BY AUTOMATICALLY CHANGING AN INDEX

Background

[0001] An embodiment of the invention described below is related to accessing software-accessible registers that are used to configure the functions of a computer system, including one or more of its integrated circuit components.

[0002] A computer system is composed of several primary integrated circuit (IC) components, typically including a processor such as a PENTIUM processor by Intel Corp., Santa Clara, California, and a main memory. In many cases, a system interface component (also referred to as a system chipset) is also used, to allow the processor to communicate with a broad range of other IC components and peripherals, such as a graphics subsystem, a network interface controller, and a mass storage device. Each IC component is designed to implement certain functions that work together to help the system achieve its overall purpose.

[0003] Of the primary IC components described above, the system chipset may have the most number of different functions to support. That is because the system chipset in many cases acts as a communications bridge between not only the processor and other IC components, but sometimes also between the peripherals and main memory. To use IC manufacturing resources more efficiently, a single system chipset is often designed to be flexible enough to support different types of processor, memory, and peripheral combinations. This flexibility is achieved by designing the system chipset with configurable, internal hardware whose configuration and functionality is set according to the contents of a number of internal, software-accessible registers (also referred to as configuration registers). After applying power to a given computer system, the configuration registers in the system chipset (and also those in other primary IC components such as the processor and memory) are written with their desired values, typically under the control of an operating system program.

[0004] In the case of the 82443BX host bridge/controller chipset by Intel Corp., the programming of a configuration register (also referred to as programming in configuration space) may be as follows. There are two control registers that are accessible from the host/ central processing unit (CPU) I/O address space, namely a Configuration Address Register (CONFADD, also referred to by its hex address CF8) and a Configuration Data Register (CONFDATA, also referred to by its hex address CFC). Access to any of the configuration registers is by first accessing the CONFADD and identifying the selected register (by appropriately writing to the CONFADD to "point" to the selected register). Next, an access is performed to the CONFDATA which contains the current value of the selected register, to write a new value. The selected register will then be updated accordingly. To program another configuration register, the same dual access operation is repeated to first point to the next register (by writing to the CONFADD), and then request an update to it (by writing the new value to the CONFDATA).

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] The embodiments of the invention are illustrated by way of example and not by way of limitation in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an" embodiment of the invention in this disclosure are not necessarily to the same embodiment, and they mean at least one.

[0006] **Fig. 1** is a flow diagram of a method for accessing a sequence of configuration registers that may accelerate the process of initializing a computer system.

[0007] **Fig. 2** shows two different methods for accessing the configuration registers, from the point of view of the "transmitter ", *e.g.* a programmed host/CPU.

[0008] **Fig. 3** illustrates three groups of configuration registers that may be programmed separately in block mode.

[0009] **Fig. 4** is a logic diagram of part of an IC component that allows for automatic indexing of configuration registers.

[0010] **Fig. 5** depicts a block diagram of a computer system in which configuration registers of a chipset component can be programmed by automatic indexing.

DETAILED DESCRIPTION

[0011] It has been found that the number of configuration registers in a computer system's primary IC component, particularly a system chipset, are likely to increase significantly for future computer architectures. For example, in a chipset that supports a system interconnect of point-to-point links (to multiple processors, other IC components, and peripherals), there may be several hundred registers to verify and/or program upon each system initialization. In addition, these registers may become wider in the future, so that an even greater amount of configuration data will need to be programmed. Finally, in most cases, the bus protocol and bus cycles traditionally defined for accessing configuration registers are relatively slow, compared to those aimed at accessing memory, for instance, because there was no perceived need for a fast configuration process. These factors together are expected to significantly slow down the process of initialization in future computer systems.

[0012] Turning now to **Fig. 1**, a flow diagram of a method for accessing a sequence of configuration registers is shown that may accelerate the process of initializing a computer system. **Fig. 1** depicts operations that may be performed by the "receiver ", for example a system chipset component in which the configuration registers and associated hardware for programming them are located. In contrast, **Fig. 2** described below refers to a method for accessing the configuration registers, from the point of view of the "transmitter", *e.g.* a programmed host/CPU. Beginning with **Fig. 1**, operation may start with receiving an indication that an attempt has been made to access an address register (operation 104). The address register may, for example, be the CONFADD at hex address CF8 mentioned above. The access may be an attempt to write a given index value that points to a given configuration register of the sequence. Alternatively, the attempt may be to read an index value from the address register. In either case, operation proceeds with block 108.

[0013] In block 108, an indication is received that an attempt has been made to access another register, referred to here as the data register. For example, this attempt may be an attempt to write a given content value that is

to be part of the content of a given configuration register. Next, operation proceeds with changing the index variable to point to another configuration register of the sequence (block 112). This is done without waiting for another attempt to access the address register. In other words, the index is automatically changed to point to another configuration register. This helps accelerate the overall process of programming the configuration registers, because it avoids the additional attempt to access the address register for each subsequent configuration register. This mode of operation for an integrated circuit (IC) component is also referred to here as a special or block mode of operation.

[0014] There are several variations to the process described above. First, one or both of the given index and content values, as they have been written to the address and data registers, respectively, may be encoded. An example will be given below in connection with the computer system shown in **Fig. 4** in which the index value is extracted or derived from a combination of several fields of an address register. In addition, the automatic changing of the index variable may be allowed, for example, only if the block mode of operation, for programming the sequence of registers via automatic indexing, has been previously entered. Examples of normal and block mode processes are shown in **Fig. 2**. Yet another variation is that the change in the index variable may be a fixed increment or a decrement so that the registers of the sequence are programmed sequentially according to ascending or descending addresses. Finally, the index variable may be implemented by a logic structure (*e.g.*, a counter or register) separate from the address register. Other variations are also possible.

[0015] The method described above for accessing a sequence of two or more configuration registers continues, as shown in **Fig. 1**, with receiving an indication of another attempt to access the data register (block 114). Next, in response to receiving the further indication in block 114, and before receiving yet another attempt to access the address register, the index variable is changed to point to yet another configuration register in the sequence (block 118). The operations in blocks 114 and 118 may be repeated as indications of further

attempts are received to access the data register, to point to still further configuration registers in the sequence.

[0016] Note that the process described above is suitable for programming, which includes new content values being actually written into the selected configuration registers (based upon the content of the data register). In addition, although the operations described above refer to receiving indications that attempts have been made to access either the address or data registers, this should be understood as including not just, for example, detecting certain types of bus events (*e.g.*, a special bus transaction aimed at the address and/or data registers), but also detecting, for example, with respect to block 104, that the address register has, in fact, been written with a new index value. Similarly, as to block 108, receiving an indication that an attempt has been made to access the data register may be that a particular bus event aimed at the data register has been detected, or that a new content value has been written to the data register. Other ways of receiving an indication that an attempt has been made to access a register may be used.

[0017] Turning now to **Fig. 2**, a flow diagram for a method of programming configuration registers is shown, from the point of view of the transmitter (*e.g.*, host/CPU). There are actually two different processes depicted, where depending upon whether the block mode has been enabled (block 202), one or the other process may be performed. In this example, the same set of configuration registers are programmed in both processes. In the process depicted on the left side of **Fig. 2** (the normal mode of operation), three configuration registers at index values 01, 02, and 03 are programmed in accordance with blocks 204-224. It should be noted that in this process, a dual stage operation is performed for each register. For example, to program the register at index value 01, blocks 204 and 208 are performed in that sequence where first the index value is written to the address register (appearing in this embodiment at hex address CF8) followed by a read or a write of the configuration data or content value to the data register (at hex address CFC).

[0018] In contrast to the process on the left of **Fig. 2**, the one on the right takes advantage of automatic indexing by essentially eliminating in this

embodiment the operations described in blocks 212 and 220, and replacing them with an automatic increment to the next sequential index (blocks 210 and 218). Where there are relatively large numbers of configuration registers to be programmed, there is a substantial savings in time associated with eliminating an entire write operation to the address register CF8.

[0019] Note that the IC component which supports the methodology depicted in **Fig. 2** may have two modes of operation for programming the configuration registers, one of which is the block mode as exemplified in the right hand flow of **Fig. 2** while the other is the normal mode shown in the left side of the figure. The IC component may thus be designed with a further register that is accessible to software, for programmably enabling or disabling the block mode of operation. This allows different groups of configuration registers to be programmed in different modes, at a substantial time savings. For example, a relatively large, first group of registers are programmed sequentially in block mode. The IC component may then be returned to normal mode for jumping to the first register of the next group (which does not follow sequentially the first group). This ability is depicted in **Fig. 3** where first the group with indexes or addresses 1-6 is programmed in block mode; then the IC component is placed in normal mode to program the register at address 21; followed by block mode to program the registers at addresses 54-59; and then normal mode is entered to program the lone register at address 104, followed by block mode to program the registers 174-180 sequentially.

[0020] Turning now to **Fig. 4**, a logic diagram of part of an IC component that allows for automatic indexing of configuration registers is shown. The IC component may be a system chipset, a processor, or other primary IC component of a computer system. A number of configuration registers 304 are provided in the IC component, for example, on-chip with some core logic of the IC component (not shown). Every one of these registers 304 may be indexed by an output of multiplexor logic 308. Once indexed or selected, some or all of the contents of the selected register (which is also referred to in the figure as "data") may be transferred to a data register (CFC, not shown) under the control of a hardware control signal CFC Read Command. Alternatively, data may be written to the selected register, under control of the hardware control

signal CFC Write Command. The CFC Write and Read Commands are examples of signals which indicate that requests have been received (*e.g.*, from outside of the IC component) to write and read, respectively, the data register (CFC).

[0021] The multiplexor logic 308 may have two inputs. A first input may receive an index value from a counter 312, while a second input may receive an index value derived directly from, in this embodiment, the address register (CF8) to which the index value has been previously written. The output of the multiplexor logic 308 is selected between the index value from the counter 312 and the index value obtained from the address register (CF8), in accordance with a block mode enable control signal which indicates a mode of operation of the IC component.

[0022] The index value provided by the counter 312 can change, by for example automatically incrementing or decrementing the counter 312. An increment or decrement signal may be provided by detection logic 316.

[0023] In operation, when the IC component is in its normal mode, the index provided to the configuration registers 304 may be derived from the contents of CF8 as it has been written by software. However, when block mode has been enabled, the index is obtained from the counter 312. Asserting the block mode enable signal allows the counter 312 to be loaded with the value obtained from CF8, which subsequently provides index values that are incremented automatically as commanded by detection logic 316. Whenever the detection logic 316 determines that a request has been received (*e.g.*, from outside of the IC component) to access a configuration register, and in particular a CFC Write or Read Command has been detected, the counter 312 is incremented to point to the next configuration register. Other logic designs for implementing the automatic indexing capability can be used.

[0024] Referring now to Fig. 5, a block diagram of a computer system in which configuration registers of a chipset component that can be programmed by automatic indexing is depicted. The chipset component in this embodiment is referred to as a I/O hub 416. The I/O hub 416 acts as a bridge between a processor and main memory combination 404 on one side and one or more I/O

components 408, 412 on the other. In this embodiment, the I/O component 408 is a network interface controller while the component 412 is a graphics controller/subsystem. The I/O hub 416 may be equipped with additional ports that can be used to communicate with further components (not shown).

[0025] In this embodiment, there are four processor-main memory combinations 404 shown, although in general there can be one or more. Each of these processor-main memory combinations is communicatively coupled to the other by way of a separate, high speed data, point-to-point link, as shown. The point-to-point link may have one or more "lanes" where each lane may be comprised of a single trace on a printed wiring board together with associated I/O buffer circuitry (not shown), used for predominantly unidirectional communications between two of the processor-main memory combinations 404. In addition, point-to-point links are also used to communicatively couple the I/O hub 416 and the IC components 408, 412. The point-to-point links may be designed to operate in accordance with the PCI Express Base Specification 1.0a (April 15, 2003).

[0026] A set of configuration registers 424 and a pair of address and data registers 420 are provided in the I/O hub 416. As an example, a configuration address register (CF8) may be defined with the following fields: port number, function number, and register number. Together, the content of these fields points, perhaps through some form of encoding, are written to a selected one of the configuration registers 424. The contents of this selected register 424 will be reflected in a configuration data register (CFC).

[0027] At least some of the confirmation registers 424 may indicate how a transaction request received on one side of the bridge that is being implemented by the I/O hub 416 is transported to another side of the bridge. The registers 424 may be designed to configure other functionality, including for example device identification, vendor identification, timers, memory control and timing, error status, scratch pad, etc. In addition, although not shown, the I/O hub 416 may also include the logic circuitry needed for implementing the automatic indexing capability described above, such as the logic design shown in **Fig. 4**.

[0028] A method for programming the registers 424 in block mode, using the system of **Fig. 5**, may be described as follows. First, a predetermined type of bus event is detected which is aimed at accessing an index variable that points to one of the registers 424. For example, the index variable may be reflected in the configuration address register (CF8) which may be mapped to a host central processing unit (CPU) I/O address space of the system, such that the CF8 is accessible by any one of the processor-main memory combinations 404. Logic in the I/O hub 416 could then be designed to monitor either of the links 405 or 407, for a bus event that is aimed at accessing the CF8.

[0029] Upon detecting such a bus event, the logic may then expect a subsequent bus event that is aimed at updating one of the registers 424 to which the index variable points. This bus event may be a transaction that targets the CFC, to read or write a content value aimed at or derived from the selected/indexed configuration register 424. Next, since the I/O hub 416 has been placed into block mode, whenever the latter bus event is detected, the logic circuitry will, without waiting for another bus event that is aimed at accessing the index variable, change the index variable to point to another one of the configuration registers 424. Note that this change in the index variable may also be reflected in the address register (CF8), where there is a subsequent read of that register. In addition, the current contents of the configuration register to which the index variable points may be reflected in the data register (CFC) when there is a subsequent read of that register. So long as block mode remains enabled, the logic circuitry may continue to change the index variable (to point to another one of the configuration registers) each time it detects a bus event aimed at accessing the data register (CFC), without requiring a bus event to update the address register (CF8).

[0030] To summarize various embodiments of a method and apparatus for accessing configuration registers by automatically changing an index have ~~been described. In the foregoing specification, the invention has been~~ described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention as set forth in the appended claims. For example, although **Fig. 5** shows the

configuration and control registers 424, 420 being in the I/O hub 416 and accessed via "frontside" links 405, 407, others may be located in the processor-memory combination 404 or even in IC components 408, 412, and accessed via a platform management system 416 through a low speed bus (not shown). The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.